공개키 암호화 프로세서에 적합한 이진 덧셈기의 구조 연구

문상국

목원대학교 전자공학과

Design of a Binary Adder Structure Suitable for Public Key Cryptography Processor

Sangook Moon

Mokwon University, Department of Electronic Engineering

E-mail : smoon@mokwon.ac.kr

요 약

현재까지 이진 덧셈기에 대한 연구는 다양한 방법으로 연구되었다. 비동기식 덧셈기들의 최악 지연시간과 평균 지연시간에 대한 연구에 의하면, 하이브리드 구조의 캐리선택 덧셈기가 리플캐리 덧셈기에 비해 32비트 비동기 RISC 프로세서에서 17%, 64비트 마이크로프로세서에서 23%의 성능 향상을 보였다. RSA와 같이 복잡하고 고성능의 연산을 필요로 하는 프로세서 시스템에서는 가장 기본적인 연산을 수행하는 덧셈기에 대한 최적화가 필수적이다. 현재까지 다양한 구조와 여러 가지 방법으로 덧셈기에 대한 면적과 지연시간에 대한 연구는 덧셈 방식이나 덧셈기 구조에 대한 것이 대부분이었다. 본 논문에서는 자동 합성 측면에서 덧셈기의 성능을 분석하고 설계하였다. 덧셈기를 소그룹으로 나누어 각 소그룹에 대한 크기 차이와 합성 방법에 따라서 구현된 덧셈기들의 성능 및 소요 면적을 분석하여 복잡한 대단위 연산을 요하는 공개키 암호화 프로세서에 적합한 최적화된 덧셈기의 구조를 제안한다.

ABSTRACT

Studies on binary adder have been variously developed. According to those studies of critical worst delay and mean delay time of asynchronous binary adders, carry select adders (CSA) based on hybrid structure showed 17% better performance than ripple carry adders (RCA) in 32 bit asynchronous processors, and 23% better than in 64 bit microprocessor implemented. In the complicated signal processing systems such as RSA, it is essential to optimize the performance of binary adders which play fundamental roles. The researches which have been studied so far were subject mostly to addition algorithms or adder structures. In this study, we analyzed and designed adders in an asp;ect of synthesis method. We divided the ways of implementing adders into groups, each of which was synthesized with different synthesis options. Also, we analyzed the variously implemented adders to evaluate the performance and area so that we can propose a different approach of designing optimal binary adders.

키워드

암호화 프로세서, 덧셈기, RSA, synthesis, CSA, RCA

Ⅰ. 서 론

정보의 바다라고 불리는 인터넷이 현대 사회에 없어서는 안될 주요한 역할을 담당하게되면서부터 더불어 반드시 필요한 기술이 네트워크상에서의 신원을 보증하고 인증할 수 있는 기술이다. 전통적으로 이러한 기술의 뒷 배경에는 암호화 기술이 동반되는데, 이 암호화 기술은 암호문을 해독할 수 있는 열쇠 (키)의 사용 방식에 따라 비밀키 방식과 공개 키 방식으로 분류된다. 비밀 키 방식은 구현이 상대적으로 간단하지만 정보를 교환하고자 하는 양자간이 사전에 키를 미리 교환하여서 안전하

게 보관해야 한다는 전제조건이 필요한 기술이며, 공개 키 방식은 이러한 키 교환을 네트워크를 통해서 안전하게 교환할 수 있게 해 주는 기술이다 [1].

공개 키 암호 알고리즘은 데이터의 암호화(encryption)에는 공개 키가 사용되고 복호화(decryption)에는 비밀 키가 사용되는 암호 시스템을 말한다. 미국 스탠퍼드 대학의 헬만 (M.H. Hellman) 등이 개발한 암호 시스템으로 기존의 정보 교환 분야에서 써 온 관용 암호 시스템 (비밀키 암호 시스템이라고도 한다)에서는 암호화의 복호화에 동일한 키가 사용되었으나, 공개 키 암호 시스템에서는 암호화 키와 복호화 키를 분리하여 정규적인 정보 교환 당사자 간에 암호화 키는 공개하고 복호화 키는 비공개로 관리한다. 이 시스템에서는 암호화 조작은 용이하고 복호화에는 방대한 조작이 필요하지만 어떤 복호화 키가 주어지면 용이하게 역변환이 가능하게 되는 일방향성 돌파구(trap door) 함수의 개념이 사용되고 있다. 공개 키 암호 시스템은 다수의 정보 교환 당사자 간의 통신에 적합하고 디지털 서명(digital signature)을 용이하게 실현할 수 있는 특징이 있다. 대표적인 것으로는 RSA 공개 키 암호 방식(RSA public key cryptosystem)이 있다 [2].

RSA의 연산은 법 곱셈 (modular multiplication)과 법 거듭제곱 연산 (modular exponentiation)이 주를 이루게 되는데 이러한 법 연산을 고속으로 처리할 수 있는 방법 중의 하나인 몽고메리 알고리즘이며, 이를 효율적으로 처리하기 위해서는 고성능의 누적곱셈 연산 알고리즘이 있어야 하며 연산의 가장 하위 계층에는 최적화된 고성능 덧셈기가 요구된다 [3]. 본 논문에서는 고성능 누적곱셈연산의 기본요소가 되며 이 연산을 가장 효율적으로 처리할 수 있는 덧셈기를 자동 합성 측면에서 파악하여 여러 가지 방법으로 합성된 결과물을 비교하여 성능을 분석하고자 한다.

Ⅱ. RSA 암호 알고리즘

RSA란 암호화와 인증을 할 수 있는 공개키 암호 시스템이다. 이것은 1977년 Ron Rivest와 Adi Shamir, Leonard Adleman에 의해서 개발되었다. RSA는 대단히 큰 정수의 인수분해가 곤란함을 기반으로 보안성과 전자서명을 제공한다. 이것은 다음과 같은 동작 원리를 가진다.

2.1. RSA의 키 생성 알고리즘

- 각각의 주체 (서로 암호문을 주고받을 대상)는 RSA 공개키 (public key)와 그에 상응하는 비밀키 (private key)를 생성해야 한다.

- 각각의 주체 A는 다음과 같은 과정을 수행해야 한다.

. 개략적으로 비슷한 크기의 큰 임의의 소수 p, q를 생성한다.

. $n = pq$ 와 $\phi = (p-1)(q-1)$를 계산한다.

. 임의의 정수 e 선택

($\gcd(e, \phi) = 1$을 만족하는 $1 < e < \phi$)

. 확장 유클리드 알고리즘을 사용하여 특별한 정수 d를 계산한다.

($ed \equiv 1 (\mathrm{mod}\, \phi)$을 만족하는 $1 < d < \phi$)

. 공개키 (n, e), 비밀키 (d)

2.2. 확장 유클리드 알고리즘 (extended Euclidean algorithm)

- INPUT : $a \geq b$ 인 두개의 음이 아닌 정수

- OUTPUT : $d = \gcd(a, b)$와 $ax + by = d$를 만족하는 정수 x, y

- RSA 키 생성 알고리즘에서 생성된 정수 e와 d를 암호화 지수 (exponent) 또는 복호화 지수라 부르고, n을 모듈러스 (modulus)라 한다.

2.3. RSA 공개키 암호화와 복호화

B가 A에게 보낼 메시지 m을 암호화하고, A는 복호화 한다.

* 암호화 (encryption) : B는 다음 과정을 수행한다.

- A의 신뢰할 수 있는 공개키 (n, e)를 가져온다.

- $[0, n-1]$ 의 간격으로 메시지를 정수 m으로 나타낸다.

- $c = m^e \bmod n$ 을 계산한다.

- 암호문 c를 A에게 보낸다.

* 복호화 (decryption) : A는 다음 과정을 수행한다.

- 암호문 c에서 평문 m을 얻기 위해 비밀키 (private key) d를 사용한다.

- $m = c^d \bmod n$을 계산한다.

2.4. RSA 복호화 과정

$ed \equiv 1 (\bmod \phi)$ 이기 때문에, $ed = 1 + k\phi$를 만족하는 정수 k가 존재한다.

페르마의 정리 (Fermat's theorem)에 의해 만약 $\gcd(m, p) = 1$ 이면 $m^{p-1} \equiv 1 (\bmod p)$ 이다.

이식의 양변에 $k(q-1)$ 제곱을 하고 , m을 곱하면 식은 다음 식 (1)과 같다.

$$m^{1+k(p-1)(q-1)} \equiv m (\bmod p) \qquad (1)$$

만약 $\gcd(m, p) = p$이면 $\bmod p$ 의 0 양변이 합동이기 때문에 위의 합동은 유효하다. 그러므로 모든 경우에 대해 $m^{ed} \equiv m (\bmod p)$ 이다.

같은 방식으로, $m^{ed} \equiv m (\bmod q)$일 때, p, q가 다른 소수이기 때문에 $m^{ed} \equiv m (\bmod n)$ 이고 $c^d \equiv (m^e)^d \equiv m (\bmod n)$ 이다.

III. 제안하는 덧셈기 방식

덧셈기는 크게 직렬, 트리 구조, 하이브리드 구조와 같이 세가지로 분류되어 왔다. 직렬 구조 덧셈기는 최대 지연 시간이 O(n)인 경우에 면적이 O(n)이 되는 특성을 지니어 시간과 면적이 비례하는 특성을 보인다. 트리 구조는 최대 지연 시간을 상당히 감소시키는 대신 면적 비용을 요구한다. 하이브리드 덧셈기는 이 둘 중 중간의 성능을 보인다. 면적이 O(n)으로 직렬 방식과 동일한 반면, 지연 시간은 $O(\sqrt{n})$이 되어 속도면에서 장점을 보인다

덧셈기는 7가지 구조를 실제로 구현해 보고 회로 자동합성기술을 사용하여 합성한 다음 성능을 비교하였다. 구현한 가산기의 방식은 리플캐리가산기 (RCA), 캐리예측가산기 (CLA), 캐리예측가산기 기반의 리플캐리가산기 (RCAon CLA), 리플캐리가산기 기반의 캐리예측가산기 (CLAonRCA), 캐리예측가산기 기반의 캐리선택가산기 (CSAonCLA), 리플캐리가산기 기반의 캐리선택가산기 (CSAonRCA) 등 7가지 방식이다. 7가지 다른 구조로 설계하고 시뮬레이션해본 결과, 타겟으로 삼은 RSA 프로세서에 가장 적합한 덧셈기는 16비트 캐리예측가산기를 기반으로 하는 리플캐리가산기로 선택되었다. 그림 1은 선택된 하이브리드 구조의 기본 구조가 되는 16비트 덧셈기의 블록도이다.

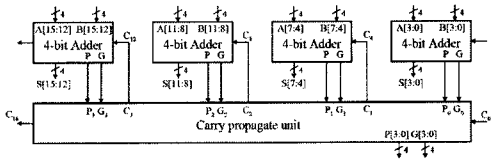

그림 1. 16비트 하이브리드 덧셈기 블록도
Fig. 1. 16-bit hybrid adder block diagram

IV. 덧셈기의 구현

덧셈기는 두가지 방식으로 자동 합성하였다. 첫번째 방식은 하위 세부 기능 블록을 grouping 한 후에 합성을 수행하였고, 두 번째 방식은 ungrouping 하여 모든 세부 모듈을 최하위 게이트 레벨로 풀어서 합성하였다. 공정은 삼성 0.25um CMOS 표준 셀 라이브러리를 사용하였고, 자동 합성 도구의 옵션에서 group을 변화시켜 합성하였다. 시뮬레이션의 최악 조건은 2.3V, 섭씨 100도이다.

각기 다른 방식으로 합성한 덧셈기는 HDL을 이용하여 탑다운 설계방식을 사용하였다. 검증은 HDL 시뮬레이터에서 택한 덧셈기의 출력과 C program으로 작성한 덧셈기의 출력이 일치하는지를 10만개의 테스트 벡터로 확인하였다.

group으로 합성한 덧셈기는 덧셈기의 일반적인 특성을 보였는데, ungroup 옵션을 사용하여 합성한 덧셈기는 지연시간 면에서 최적화된 특성을 보였다.

합성 결과에서 ungroup은 추가적인 게이트를 최대 115% 소비하여 딜레이를 최대 94% 감소시키는 결과를 보였다. 특히 RCA와 CLAonRCA에서의 지연시간에 대한 이득은 80%에서 93%까지 얻을 수가 있었다.

반면, CSA를 사용한 덧셈기는 ungroup을 했음에도 불구하고 면적을 손해보지 않으며, 추가적으로 지연시간에 대한 이득까지 보였다. 특히 소그룹을 4비트로 나누었을 때 지연시간에서 큰 폭의 이득을 얻을 수 있었다.

그림 2와 그림 3은 각각의 방식을 달리하여 합

성한 덧셈기들의 지연시간과 면적을 비교한 성능 차이를 보인다.

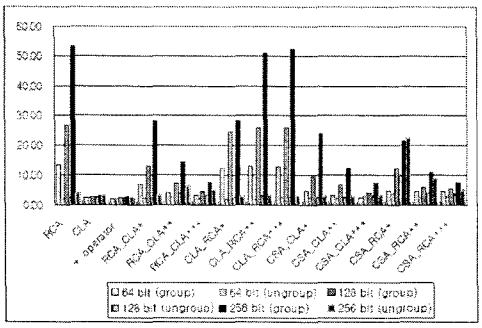

그림 2. 입력 비트 수에 따른 덧셈기의 지연시간 비교

Fig. 2. Adder delay comparison depending on the input bits

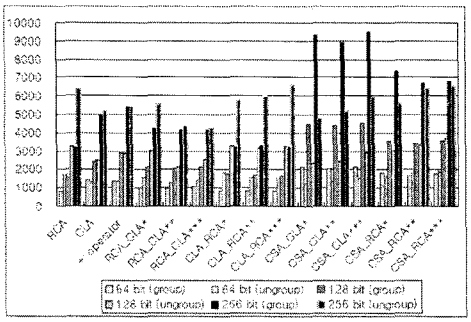

그림 3. 입력 비트 수에 따른 덧셈기의 면적 비교

Fig. 3. Adder area comparison depending on the input bits

V. 결 론

본 연구에서 제시한 합성 방식에 따른 최적화된 덧셈기의 구조를 이끌어내는 방법은 스마트 카드 등 복잡한 정보보 연산을 요하는 암호화 프로세서에 사용이 적합하다. 암호화 프로세서의 프로세서 코어가 한 번 실행될 때 내부적인 연산장치의 덧셈기는 곱셈 블록을 얼마의 단위로 잡느냐에 따라서 수 번을 수행해야 하기 때문에 주파수 성능도 필요하고, 또한 단가와 이동성을 위해 면적 또한 작아야 한다. 논 연구의 실험 결과로, 암호화 프로세서에 적합한 32비트 덧셈기로는 ungroup으로 합성된 16비트 CLA를 기반으로 하는 256비트 RCA가 선택되었다. 최적화한 덧셈기는 200MHz에서 정상적으로 동작하며 기본 게이트 기준으로 약 4500의 면적을 차지한다.

참고문헌

[1] 김철, 암호학의 이해, 영풍문고, 1996.

[2] R. L. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," Communications of the ACM, Vol. 21, pp. 120-126, Feb. 1978.

[3] 허석원, "스마트카드 구현에 적합한 최적화된 RSA 암호화프로세서 설계", 연세대학교 석사 학위 졸업논문, 2003.